AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Original) A method for reducing the overhead involved in executing native code methods in an application running on a virtual machine, comprising:

selecting a call to a native code method to be optimized within the virtual machine;

decompiling at least part of the native code method into an intermediate representation;

obtaining an intermediate representation associated with the application running on the virtual machine which interacts with the native code method;

combining the intermediate representation for the native code method with the intermediate representation associated with the application running on the virtual machine to form a combined intermediate representation; and

generating native code from the combined intermediate representation, wherein the native code generation process optimizes interactions between the application running on the virtual machine and the native code method.

2. (Original) The method of claim 1, wherein selecting the call to the native code method involves selecting the call based upon at least one of:

the execution frequency of the call; and the overhead involved in performing the call to the native code method as compared against the amount of work performed by the native code method.

3. (Original) The method of claim 1, wherein optimizing interactions between the application running on the virtual machine and the native code method involves optimizing calls to the native code method by the application.

4. (Original) The method of claim 1, wherein optimizing interactions between the application running on the virtual machine and the native code method involves optimizing callbacks by the native code method into the virtual machine.

5. (Original) The method of claim 4, wherein optimizing callbacks by the native code method into the virtual machine involves optimizing callbacks that access heap objects within the virtual machine.

6. (Currently amended) The method of claim 4,

wherein the virtual machine is a platform-independent virtual machine ~~Java Virtual Machine (JVM)~~; and

wherein combining the intermediate representation for the native code method with the intermediate representation associated with the application running on the virtual machine involves integrating calls provided by an interface for accessing native code ~~the Java Native Interface (JNI)~~ into the native code method.

7. (Original) The method of claim 1, wherein obtaining the intermediate representation associated with the application running on the virtual machine involves recompiling a corresponding portion of the application.

8. (Original) The method of claim 1, wherein obtaining the intermediate representation associated the application running on the virtual machine involves accessing a previously generated intermediate representation associated with the application running on the virtual machine.

9. (Original) The method of claim 1, wherein prior to decompiling the native code method, the method further comprises setting up a context for the decompilation by:

determining a signature of the call to the native code method; and

determining a mapping from arguments of the call to corresponding locations in a native application binary interface (ABI).

10. (Original) A computer-readable storage medium storing instructions that when executed by a computer cause the computer to perform a method for reducing the overhead involved in executing native code methods in an application running on a virtual machine, the method comprising:

selecting a call to a native code method to be optimized within the virtual machine;

decompiling at least part of the native code method into an intermediate representation;

obtaining an intermediate representation associated with the application running on the virtual machine which interacts with the native code method;

combining the intermediate representation for the native code method with the intermediate representation associated with the application running on the virtual machine to form a combined intermediate representation; and

generating native code from the combined intermediate representation, wherein the native code generation process optimizes interactions between the application running on the virtual machine and the native code method.

11. (Original) The computer-readable storage medium of claim 10, wherein selecting the call to the native code method involves selecting the call based upon at least one of:

the execution frequency of the call; and

the overhead involved in performing the call to the native code method as compared against the amount of work performed by the native code method.

12. (Original) The computer-readable storage medium of claim 10, wherein optimizing interactions between the application running on the virtual machine and the native code method involves optimizing calls to the native code method by the application.

13. (Original) The computer-readable storage medium of claim 10, wherein optimizing interactions between the application running on the virtual machine and the native code method involves optimizing callbacks by the native code method into the virtual machine.

14. (Original) The computer-readable storage medium of claim 13, wherein optimizing callbacks by the native code method into the virtual machine involves optimizing callbacks that access heap objects within the virtual machine.

15. (Currently amended) The computer-readable storage medium of claim 13,

wherein the virtual machine is a platform-independent virtual machine~~Java Virtual Machine (JVM)~~; and

wherein combining the intermediate representation for the native code method with the intermediate representation associated with the application running on the virtual machine involves integrating calls provided by an interface for accessing native code ~~the Java Native Interface (JNI)~~ into the native code method.

16. (Original) The computer-readable storage medium of claim 10, wherein obtaining the intermediate representation associated with the application running on the virtual machine involves recompiling a corresponding portion of the application.

17. (Original) The computer-readable storage medium of claim 10, wherein obtaining the intermediate representation associated the application running on the virtual machine involves accessing a previously generated intermediate representation associated with the application running on the virtual machine.

18. (Original) The computer-readable storage medium of claim 10, wherein prior to decompiling the native code method, the method further comprises setting up a context for the decompilation by:

determining a signature of the call to the native code method; and determining a mapping from arguments of the call to corresponding locations in a native application binary interface (ABI).

19-27. (Cancelled)

28. (Original) A method for reducing the overhead involved in executing native code methods in an application running on a virtual machine, comprising:

deciding to optimize a callback by a native code method into the virtual machine;

decompiling at least part of the native code method into an intermediate representation;

obtaining an intermediate representation associated with the application running on the virtual machine which interacts with the native code method;

combining the intermediate representation for the native code method with the intermediate representation associated with the application running on the virtual machine to form a combined intermediate representation; and

generating native code from the combined intermediate representation, wherein the native code generation process optimizes the callback by the native code method into the virtual machine.

29. (Original) The method of claim 28, wherein the native code generation process also optimizes calls to the native code method by the application.

30. (Original) The method of claim 28, wherein optimizing the callback by the native code method into the virtual machine involves optimizing a callback that accesses a heap object within the virtual machine.

31. (Currently amended) The method of claim 28,

wherein the virtual machine is a platform-independent virtual machine~~Java Virtual Machine (JVM)~~; and

wherein combining the intermediate representation for the native code method with the intermediate representation associated with the application running on the virtual machine involves integrating calls provided by an interface for accessing native code ~~the Java Native Interface (JNI)~~ into the native code method.

32. (Original) A computer-readable storage medium storing instructions that when executed by a computer cause the computer to perform a method for reducing the overhead involved in executing native code methods in an application running on a virtual machine, the method comprising:

deciding to optimize a callback by a native code method into the virtual machine;

decompiling at least part of the native code method into an intermediate representation;

obtaining an intermediate representation associated with the application running on the virtual machine which interacts with the native code method;

combining the intermediate representation for the native code method with the intermediate representation associated with the application running on the virtual machine to form a combined intermediate representation; and generating native code from the combined intermediate representation, wherein the native code generation process optimizes the callback by the native code method into the virtual machine.

33. (Original) The computer-readable storage medium of claim 32, wherein the native code generation process also optimizes calls to the native code method by the application.

34. (Original) The computer-readable storage medium of claim 32, wherein optimizing the callback by the native code method into the virtual machine involves optimizing a callback that accesses a heap object within the virtual machine.

35. (Currently amended) The computer-readable storage medium of claim 32,

wherein the virtual machine is a platform-independent virtual machine~~Java Virtual Machine (JVM)~~; and

wherein combining the intermediate representation for the native code method with the intermediate representation associated with the application running on the virtual machine involves integrating calls provided by an interface for accessing native code ~~the Java Native Interface (JNI)~~ into the native code method.

36-39. (Cancelled)